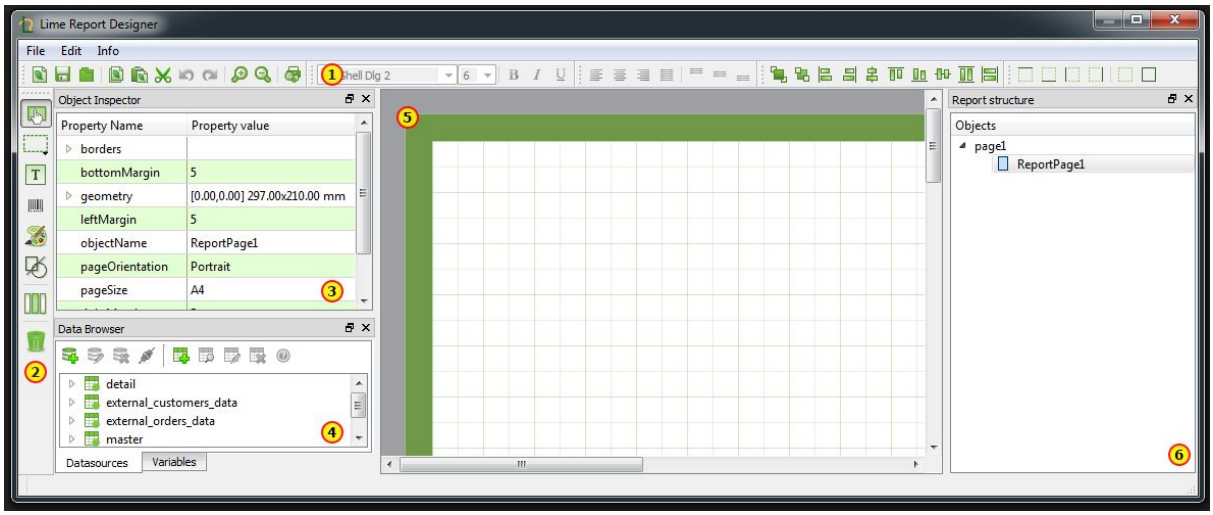
DESIGNER



1. **Toolbars**
2. **Report Elements Bar**
3. **Object Browser**
4. **Data Browser**
5. **Report Page**
6. **Report Structure Browser**

CONTROL KEYS

Ctrl+N	New Report
Ctrl+O	Load Report
Ctrl+S	Save Report
Ctrl+Shift+S	Save Report As
Ctrl+P	Render Report
Ctrl+Z	Undo
Ctrl+Shift+Z	Redo
Ctrl+C	Copy
Ctrl+X	Cut
Ctrl+V	Paste
Ctrl+Arrows	Move selected objects
Shift+Arrows	Modify size of selected objects
Del	Delete selected objects
Shift+Left mouse button	Create selection area

TOOLBARS

Main

	New report
	Save report
	Load report
	Copy selected
	Paste
	Cut
	Undo
	Redo
	Zoom in
	Zoom out
	Report preview

Font toolbar

Arial	Family
10	Size
B	Bold
I	Italic
U	Underline

Text Alignment

	Left align
	Center
	Right align
	Justify
	Top align
	Vertical center
	Bottom align

Objects Layout

	Move selected objects forward
	Move selected objects backward
	Align selected objects left
	Align selected objects right
	Align selected objects vertical center
	Align selected objects top
	Align selected objects bottom
	Align selected objects horizontal center
	Make selected objects of the same height
	Make selected objects of the same width

Borders

	Add top border to selected objects
	Add bottom border to selected objects
	Add left border to selected objects
	Add right border to selected objects
	Remove borders
	Add all borders

Report objects

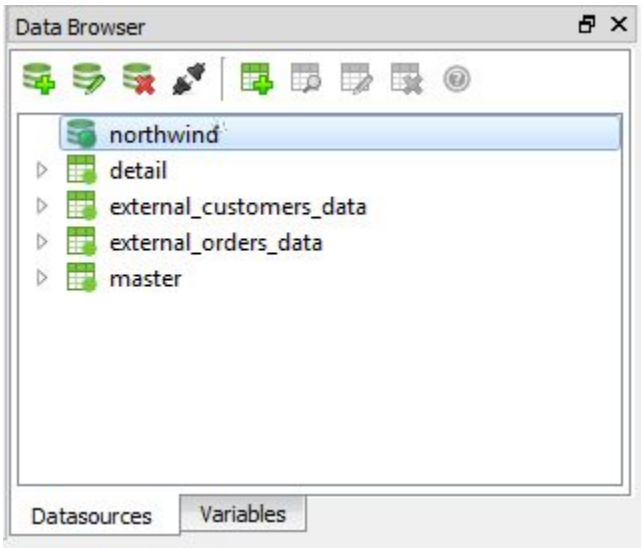
	Select an object
	Insert a band
	Insert a text
	Insert bar-code (Requires Zint library)
	Insert a picture
	Insert a geometric object
	Merge selected objects into a horizontal layout
	Remove object

DATA SOURCES

LimeReport provides several data sources:

- Variables, declared in a report and available for an external application.
- SQL based data sets using a database connection. A database connection can be established:
  - immediately from a report generator
  - by an external application.
- External datasets by implementing QAbstractItemModel.
- Data transfer and linking to a report generator by SIGNAL-SLOT approach.

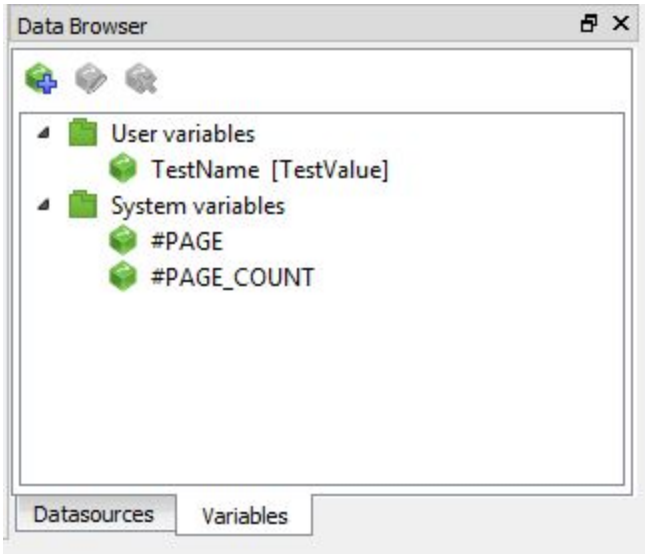
“Data Browser” window



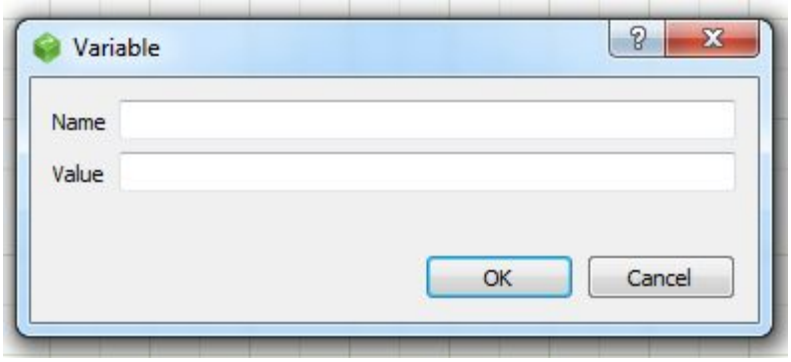
This window allows to manage data sources. One can specify parameters of connections, variables and datasets that will be used afterwards by a report building process.

Variable declaration

Variable parameters can be set on “Variables” tab of “Data Browser” window.



To add a variable push button, then in a “Variable” dialog one can specify its name and value (value can be changed by external application)

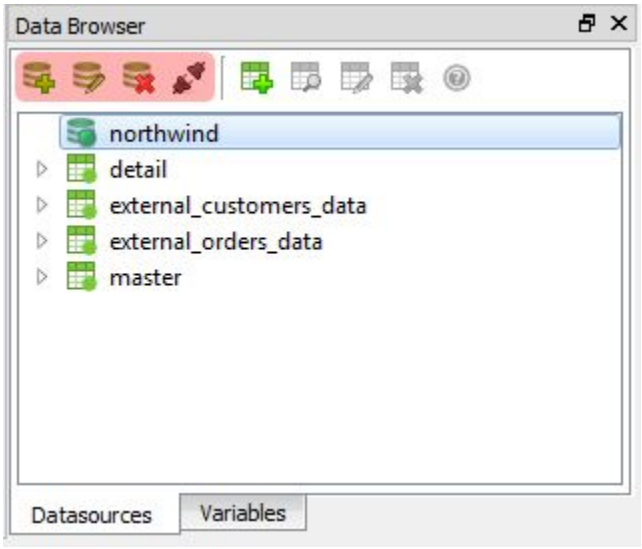


A variable declaration can be edited using button .

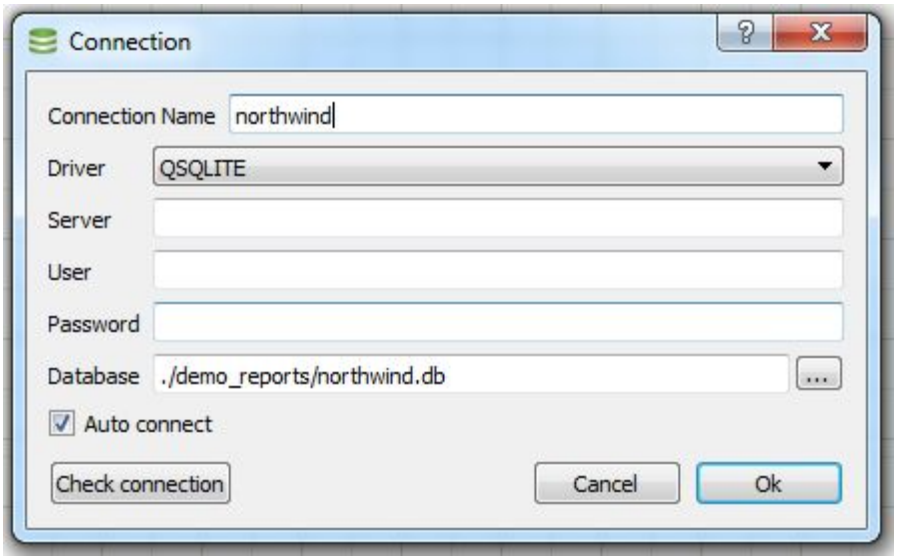
Button allows to delete a variable.

Database connection from inside report generator

Database connection parameters are set using dialog window “Connection”, by pushing button (Add database connection), located on a bar of window “Data Browser”.



“Connection” dialog



Using this dialog one can adjust connection parameters, such as:

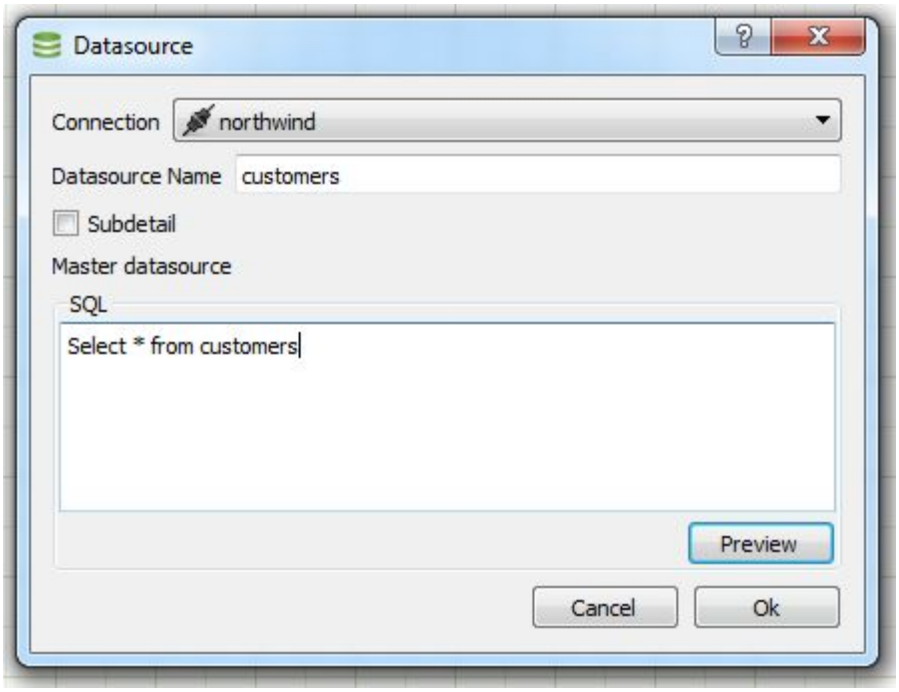
Connection Name - identifies connection in a SQL query
Driver - selects database driver
Server - server address
User - user name
Password - password
Database - database name
Auto connect - whether to connect immediately after report load. Otherwise connection will be established during report generation or can be forced by button .
A button “Check connection” allows to check parameters by establishing connection using specified parameters and shows a result message.

Connection parameters can be changed afterwards using button . To delete connection use button .

Creating datasets from report

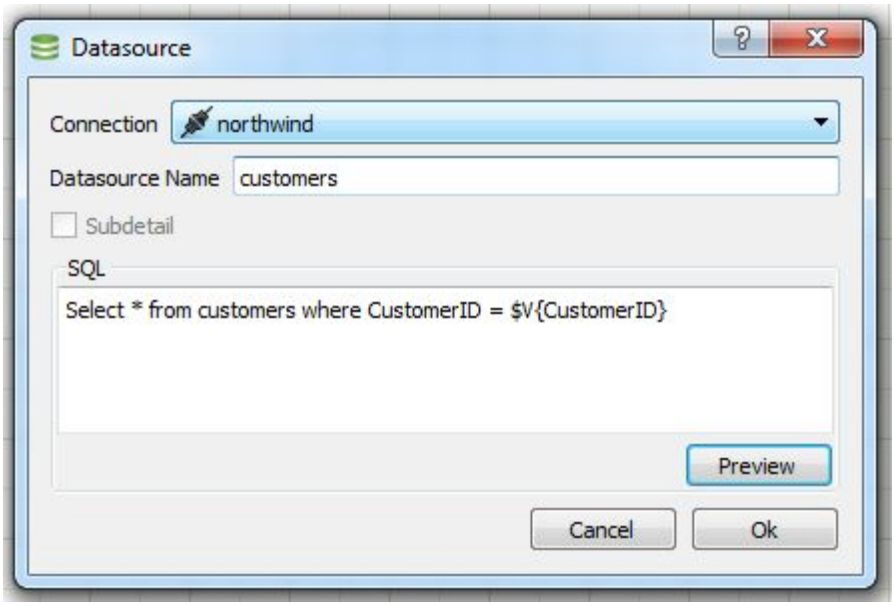
Dataset parameters can be set using “Datasource” dialog window, by pushing button , located on a bar of “Data Browser” window.

“Datasource” Dialog



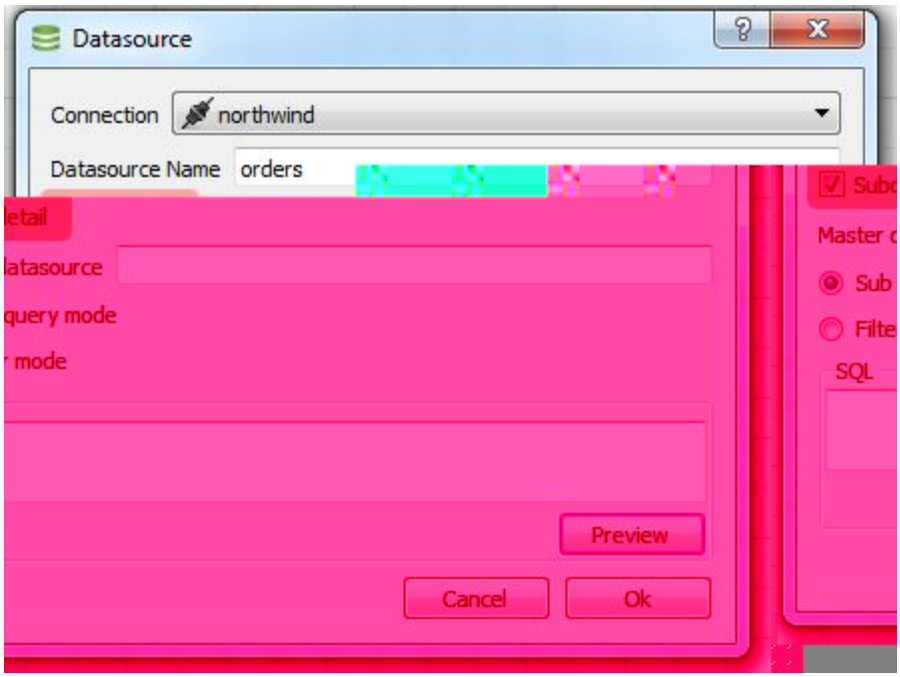
This dialog allows to declare data sources of several types:

1. Data source based on a SQL database request



An ordinary DB request can be independent as well as with parameters coming from report variables by using $V{VarName} syntax.

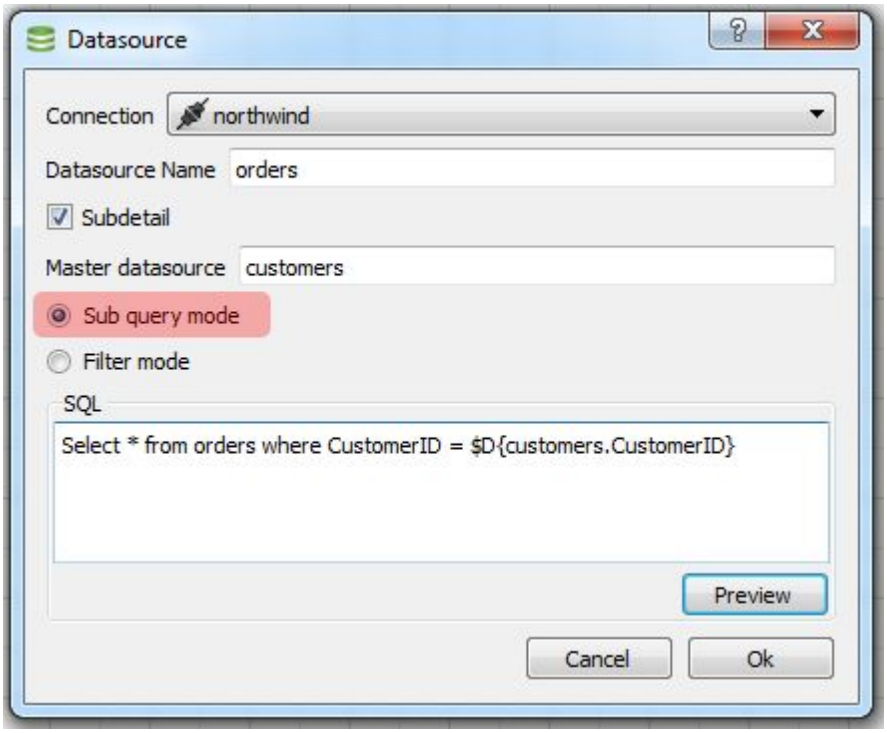
2. Secondary (linked) dataset



Secondary dataset can use another dataset as a master (parameter) During report generation data traversal goes along with secondary dataset refresh. In order to enable secondary dataset mode, check “Subdetail” in “Datasource” window.

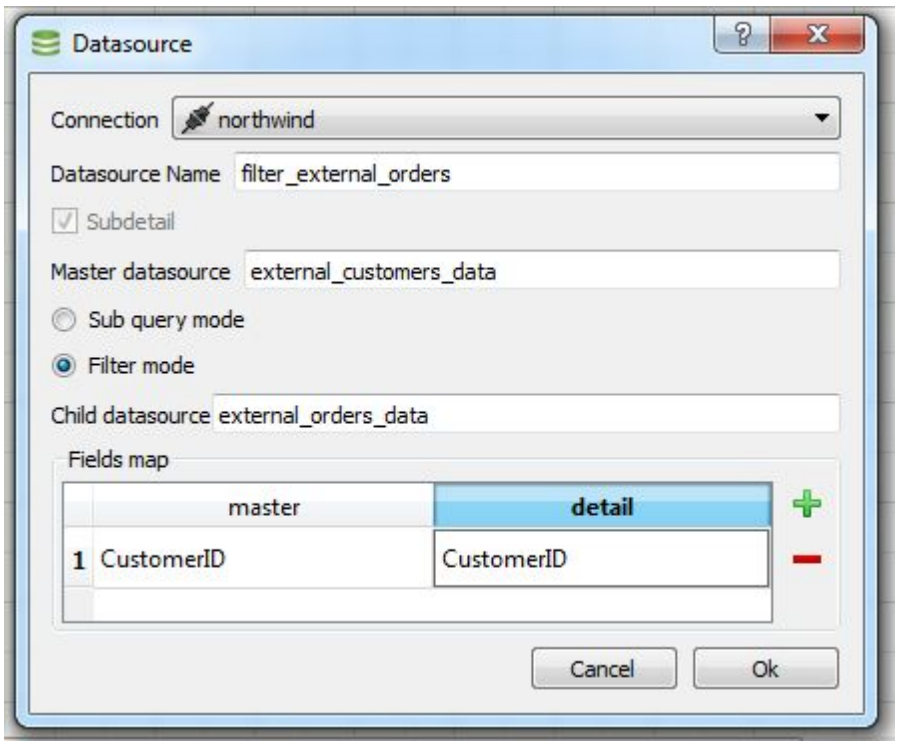
Secondary dataset can be of two types:

2.1. Secondary SQL request



To enable it check “Sub query mode”. Syntax $D{DatasetName.FieldName} is used to refer master dataset fields.

2.2. Filtering dataset



????? A filtering dataset is used when master dataset traversal results in filtering records of secondary dataset.

External data sources

External data sources can be of two types:

1. Data source implementing QAbstractDataModel interface. For example, QStringListModel. To hook up such a data source just make a call:
report->dataManager()->addModel(QString name, QAbstractDataModel* model, bool owned):

Example:

```
QStringList simpleData;
simpleData << "value1" << "value2" << "value3";
QStringListModel* stringListModel = new QStringListModel();
stringListModel->setStringList(simpleData);
report->dataManager()->addModel("string_list",stringListModel,true);
```

2. Data source hooked up using SIGNAL-SLOT mechanism.
To use this approach one should:

1. Implement receiver method (slot) (LimeReport::CallbackInfo info, QVariant &data), where info parameter has a field “dataType” (info.dataType) determining which data this method should return via “data” parameter:
    a. LimeReport::CallbackInfo::IsEmpty - data (bool) - whether source has data
    b. LimeReport::CallbackInfo::HasNext - data(bool) - whether source has next record
    c. LimeReport::CallbackInfo::ColumnHeaderData - data(QString) - index column name, specified as info.index - this value will be used afterwards to identify this column in a report and in a method to get data.
    d. LimeReport::CallbackInfo::ColumnData - data (QVariant) - value for a column with name, specified as info.columnName (this value has been set previously via a method call with a parameter LimeReport::CallbackInfo::ColumnHeaderData)
2. Implement traversal method (slot) with parameters (const LimeReport::CallbackInfo::ChangePosType &type, bool &result). A parameter “type” specifies a type of a traversal and a changing parameter “result” should be set to true if a shift has been successful, and false otherwise. Parameter “type” can be:
    a. First - move to a first record of dataset
    b. Next - move to a next record
3. Create new (ICallbackDatasource) data source using a call report->dataManager()->createCallbackDatasouce() and hook up previously created methods to signals: getCallbackData - first method, changePos - second method.

Example:

```
class MainWindow : public QMainWindow
{
…
QSqlQuery* m_customers;
…..
}

void MainWindow::prepareData(QSqlQuery* ds, LimeReport::CallbackInfo info, QVariant &data)
{
        switch (info.dataType) {
```

```
            case LimeReport::CallbackInfo::IsEmpty:
                data = !ds->first();
                break;
            case LimeReport::CallbackInfo::HasNext:
                data = ds->next();
                if (data.toBool()) ds->previous();
                break;
            case LimeReport::CallbackInfo::ColumnHeaderData:
                if (info.index < ds->record().count())
                data = ds->record().fieldName(info.index);
                break;
            case LimeReport::CallbackInfo::ColumnData:
                data = ds->value(info.columnName);
                break;
        }
}


void MainWindow::slotGetCallbackData(LimeReport::CallbackInfo info, QVariant &data)
{
        if (!m_customers) return;
        prepareData(m_customers, info, data);
}


void MainWindow::slotChangePos(const LimeReport::CallbackInfo::ChangePosType &type, bool &result)
{
        QSqlQuery* ds = m_customers;
        if (!ds) return;
        if (type == LimeReport::CallbackInfo::First) result = ds->first();
        else result = ds->next();
// В этом методе может быть реализовано обновление зависимого источника данных
        m_orders->bindValue(":id",m_customers->value("CustomerID"));
        m_orders->exec();
}

{
…
LimeReport::ICallbackDatasource * callbackDatasource = report->dataManager()->createCallbackDatasouce();
connect(callbackDatasource, SIGNAL(getCallbackData(LimeReport::CallbackInfo,QVariant&)),
        this, SLOT(slotGetCallbackData(LimeReport::CallbackInfo,QVariant&)));
connect(callbackDatasource, SIGNAL(changePos(const LimeReport::CallbackInfo::ChangePosType&,bool&)),
        this, SLOT(slotChangePos(const LimeReport::CallbackInfo::ChangePosType&,bool&)));
report->dataManager()->addCallbackDatasource(callbackDatasource,"master");
…
}
```

REPORT ELEMENTS

1. **- BAND**

"Container" element is intended to layout other report elements. A band can be of several typed:

1. Report Header - report header
2. Report Footer - report footer
3. Page Header - page header
4. Page Footer - page footer
5. Data - report data
6. SubDetail - secondary report data
7. SubDetailHeader - secondary data header
8. SubDetailFooter - secondary data footer
9. GroupHeader - group header
10. GroupFooter - group footer

Properties common for all bands:

autoHeight	Automatic height adjustment
backgroundColor	background color
borders	borders
geometry	sizes
keepBottomSpace	keep bottom margin
objectName	object name
printIfEmpty	print is there is no data on a band
splittable	Split a band if a page can not hold it

Report Footer

maxScalePercent	Maximum percent which can be used to shrink a band if it does not fit into a page. If even after shrinking the band does not fit into a page it will be moved entirely or in part depending on the band parameters.

Data

datasource	Data source. A data band will be generated for each data source record
keepFooterTogether	If Report Footer does not fit into a page, it will be moved to a next page along with a last Data band instance
sliceLastRow	Specifies whether a last Data band instance can be split or it should be moved as a whole

SubDetail

datasource	Data source. A SubDetail band will be generated for each record of a data source

SubDetailHeader, SubDetailFooter

printAlways	Print even if SubDetail is empty

GroupHeader

groupFieldName	Field by which grouping is performed. GroupHeader instance will be generated if value of this field changes

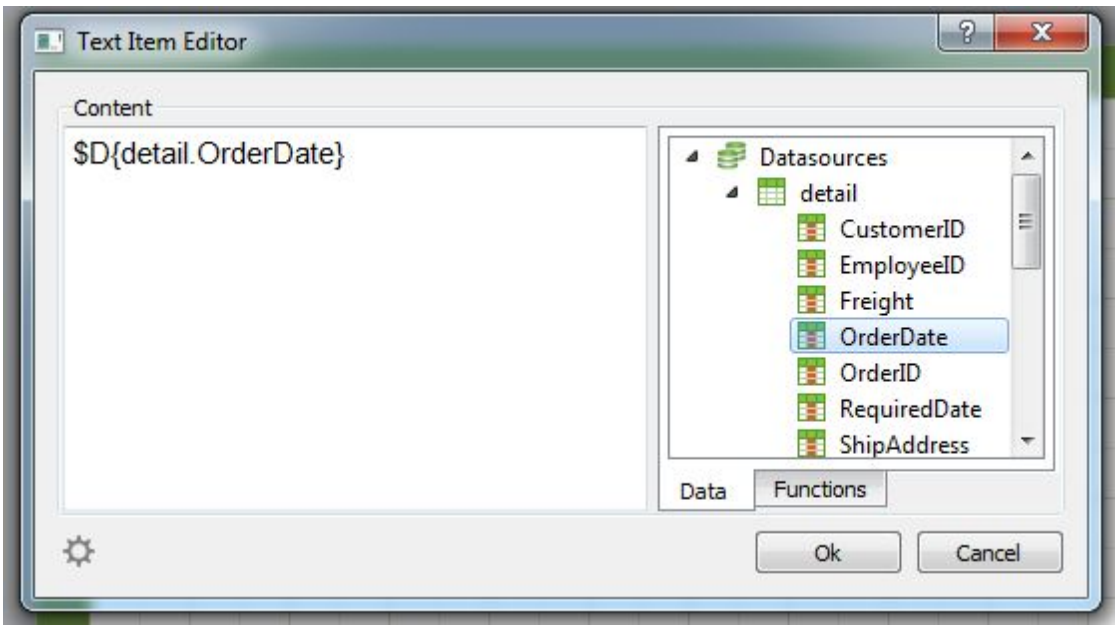
2. - **TEXT**

A text element allows to print captions and field contents of data sources.

Text parameters

alignment	Vertical and horizontal text alignment
angle	Caption rotation
autoHeight	Automatic height adjustment
autoWidth	Automatic width adjustment
backgroundColor	Background color
backgroundMode	Background filling mode
backgroundOpacity	Background filling opacity
borders	Borders
content	Text object contents
font	Font
fontColor	Font color
foregroundOpacity	Foreground opacity
geometry	Object size and layout
itemLocation	Object location (Page or Band)
margin	Margins
objectName	Object name
stretchToMaxHeight	Stretch the object so that its bottom reaches the bottom of the band in which it is placed
trimValue	Trim white characters at the ends of a text

To edit parameter “content” a “Text Item Editor” is used. To start it double click a text element.



“Content” can contain: static text, variable values, dataset field contents as well as values generated by a script execution. To show a variable value use $V{VarName} syntax, to show data field value $D{DatasetName.FieldName}, to run a script $S{ScriptBody}

3. - **BARCODE**

Element to show barcodes

angle	Rotation angle
backgroundColor	Background color
barcodeType	Barcode type
barcodeWidth	parameter for QZint
borders	Borders
content	Value to be displayed by barcode
foregroundColor	barcode color
geometry	size and layout
itemLocation	Object location (Page or Band)
objectName	Object name
pdf417CodeWords	QZint parameter
securityLevel	QZint parameter
stretchToMaxHeight	Stretch the object so that its bottom reaches the bottom of the band in which it is placed

testValue	Development time barcode value
whitespace	Whitespace

4. - **IMAGE**
Element to show images

autoSize	Adjust size to an image size
borders	Borders
datasource	Data source
field	Data field
geometry	Size and layout
image	Image
itemLocation	Object location (Page or Band)
objectName	Object name
stretchToMaxHeight	Stretch the object so that its bottom reaches the bottom of the band in which it is placed

5. - **SHAPE**
Element to draw shapes

borders	Borders
geometry	Size and layout
itemLocation	Object location (Page or Band)
lineWidth	Line width
objectName	Object name
opacity	Opacity
penStyle	Line style
shape	Shape
shapeBrush	Shape Brush
shapeBrushColor	Shape Brush Color
shapeColor	Shape Color
stretchToMaxHeight	Stretch the object so that its bottom reaches the bottom of the band in which it is placed

6. - **HORIZONTAL LAYOUT**

Element to group report items

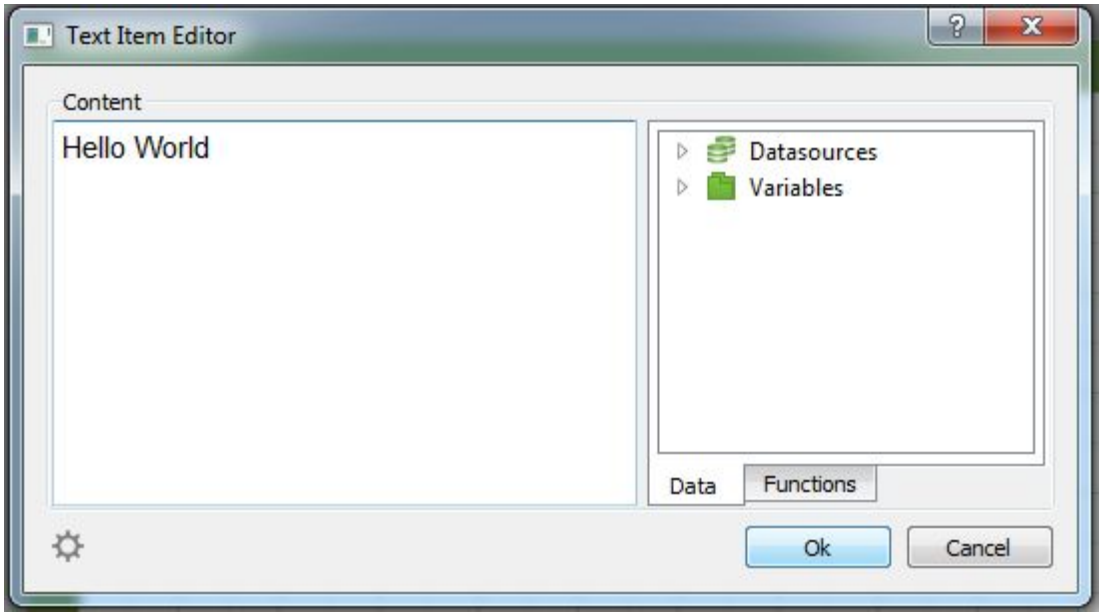
borders	Borders
geometry	Size and layout
itemLocation	Object location (Page or Band)
objectName	Object name
stretchToMaxHeight	Stretch the object so that its bottom reaches the bottom of the band in which it is placed

REPORT DESIGN

2.1. Hello World

By tradition “Hello World” is the first and the simplest one.

Create a new report . Click “Text” object on an object bar.
Move a mouse cursor to a necessary point on a page and click the mouse again.
Double click just created object thus activating an editor window.



Print text “Hello World” and push OK

The report is ready. To display it one can use preview button on a tool bar or menu item File | Render Report, one can also use Ctrl+P. A preview window with one page will appear which displays text "Hello World!".

2.2. Object text .

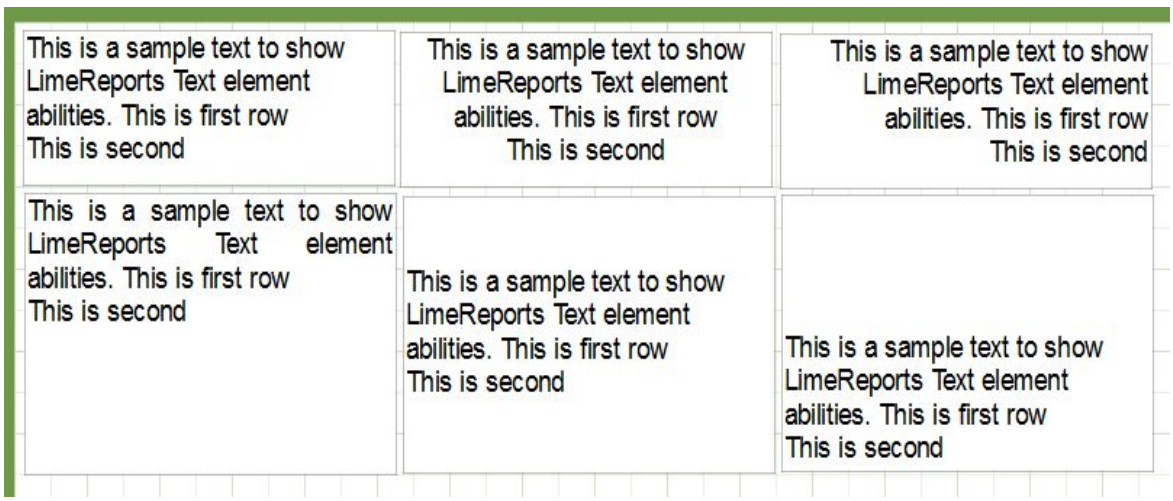
Object "Text" has powerful capabilities. It can display text, border, background. Text can be displayed by any font of any size, color and style. Most adjustments are performed visually using toolbar.

Text format examples

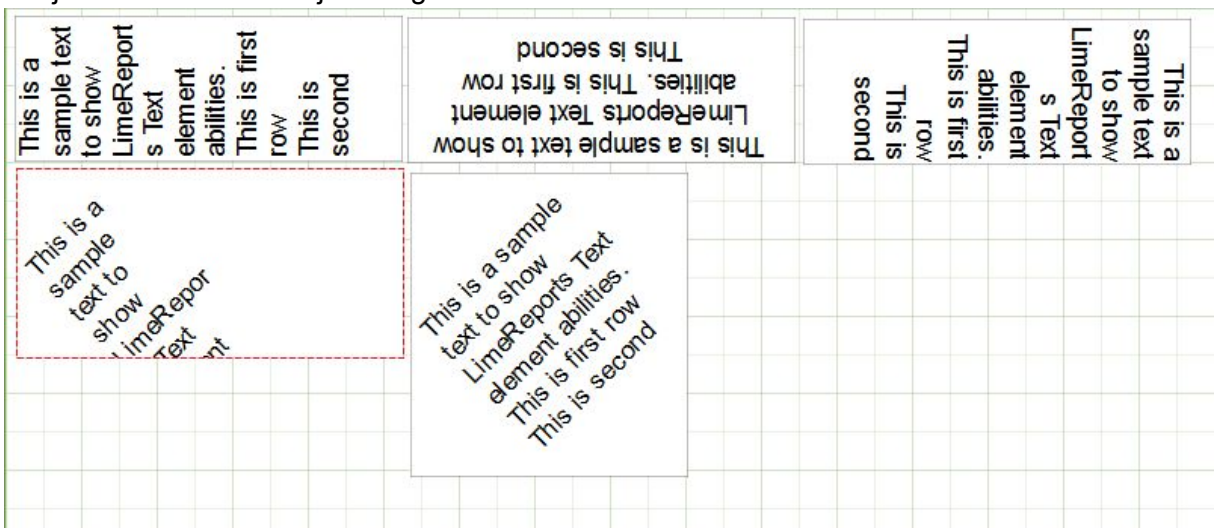
To get to know “Text” object let’s create it and put some text in it:

Let’s have a look at how alignment works inside of an object.

Alignment buttons are on a toolbar and allow independently set a horizontal and vertical alignment. Put your attention to a button "Justify" - it allows to align a paragraph on both object sides.



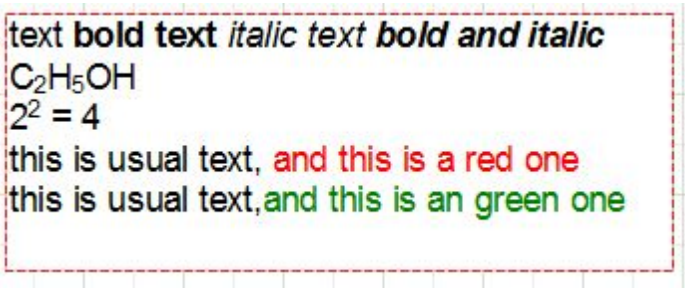
Text can be rotated by 45, 90, 180, 270 and 315 degrees. While rotating by values different from 90, 180, 270 text can overrun object edges, as shown on a picture below. So that text could fit into an object let's increase object height.



"Text" object understands HTML tags

For example:

```
text <b>bold text</b> <i>italic text <b>bold and italic</i></b>
C<sub>2</sub>H<sub>5</sub>OH
2<sup>2</sup> = 4
this is a usual text,<font color=red> and this is a red one</font>
this is a usual text,<font color = green>and this is a green one</font>
```



Displaying expressions using "Text" object.

One of the most useful capabilities of this object is to display not only a static text but also expressions. Moreover expressions can be intertwined with a static text. Let's look at a simple example - let's put into the object the following line:

After building the report, we will see approximately the following:

What has happened? While building the report LimeReport ran across an expression in the text, enclosed by $S{}, evaluated it and inserted a calculated value into the text, removing of course

auxiliary elements “$S{”, “}”. "Text" object can contain any number of expressions, mixed with an usual text. One can enclose simple variables as well as expressions, for example, $S{1+2*(3+4)}. Expressions can include constants, variables ($V{}), functions, DB fields ($D{}).

2.3. Band usage

Bands are used for logical grouping of objects. Putting an object on a "Page Header" band, we thus specify so that LimeReport displays given object at the top of every page. Similarly "Page Footer" band is displayed at the bottom of every page, along with all the objects it contains. Let’s demonstrate this by small example. Let’s make a report which contains text "Hello!" at the top of a page, current date top right and page number bottom right.

To this end click button "Insert Band" and from a poped up list select "Page Header". We see that a new band has been added to the page. LimeReport automatically layouts bands on a page in such a way so that band-headers are at the top, after them - data-bands, and band-footers below all.

Next we insert “Report Header” and “Page Footer” bands.

Now we place objects. “Text” object goes to "Page Header" band and we print $S{now()} in its editor. “Text” object with text "Hello!" goes to a "Report Header" band. And on a "Page Footer" band we place text object which has a system variable with page number $V{#PAGE}

Let’s run report and find that objects in a built report are located as necessary.

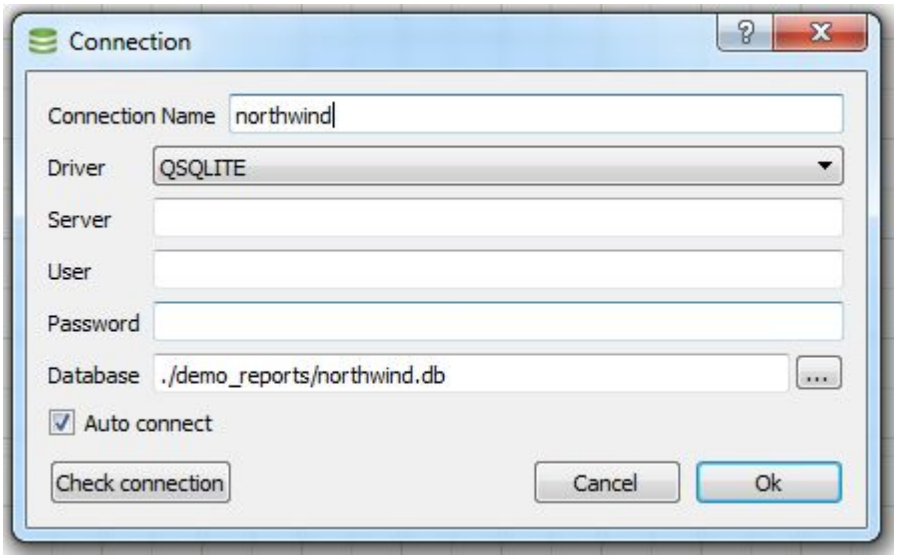
So bands are in charge of object location in a proper place. Depending on a band type we can place an object at the top or bottom of a page, on a first page or a last page. Most usefull bands work as follows: "Page Header" band is displayed at the top of every page; "Page Footer" band is displayed at the bottom of every page; "Report Header" band is displayed at the top of a first page, but lower than "Page Header" band. "Report Footer" is displayed at the end of the report on a free space.

2.4 Data-bands

Next we consider options to print data from datasets. What is a dataset in this case? This is an unknown beforehand number of records each one containing a definite number of columns (fields). To print such an information LimeReport uses a special band type - data-band. These are bands named "Data" and “SubDetail”. In order to print a whole table or some its fields, it is necessary: - add data-band to a report; - link it to a table; - put “Text” objects with fields we want to print. While building a report LimeReport repeats band as many times as many records our table has. At that if there is no free space left on a page, new report pages will be generated.

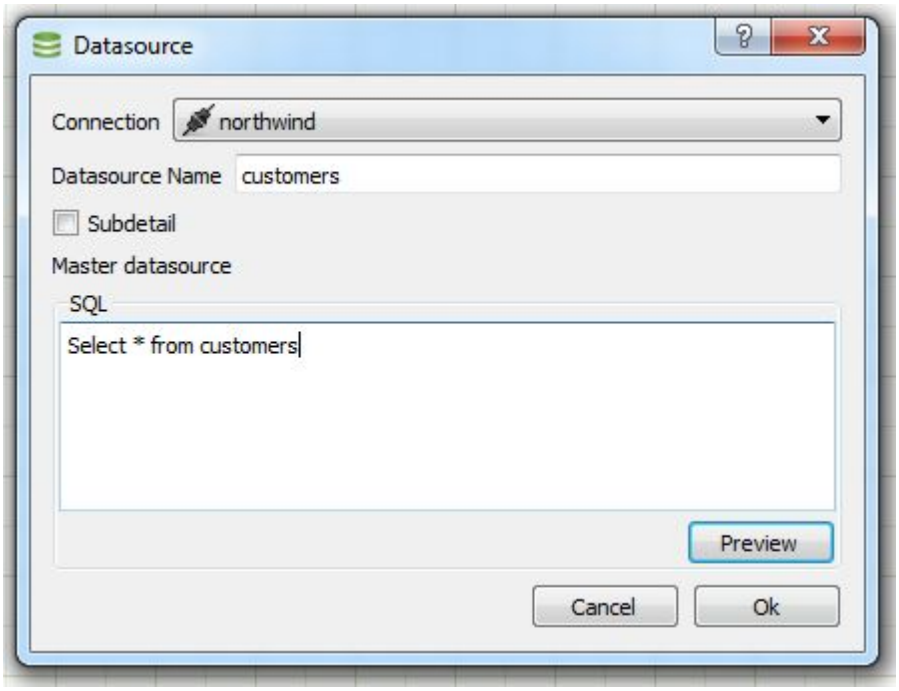
2.5 "Client list" report

This report will contain data from a clients table from demonstration DB “northwind”. For a start push a button to create new report. Next we create connection to DB. To this end we use button (Add database connection), located on toolbar of “Data Browser” window. In a popped up dialog “Connection” let’s enter connection parameters.



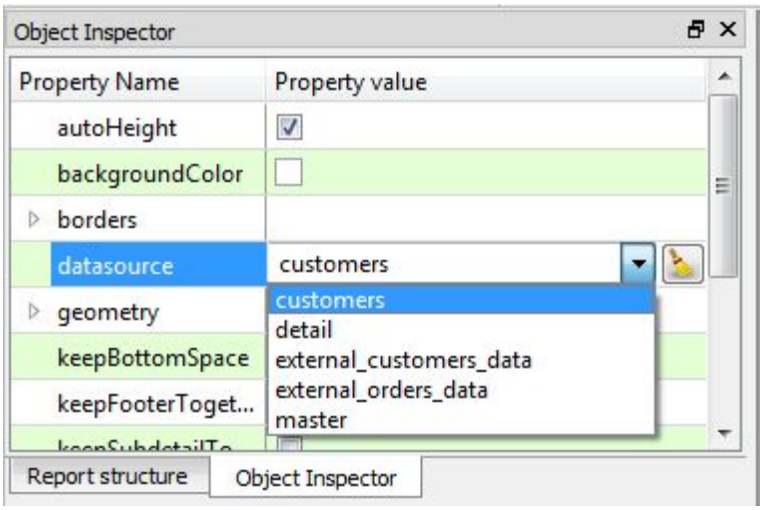
Connection Name - connection name
Driver - database driver (SQLite)
DataBase - database path
AutoConnect - automatically connect after creating connection description and report loading.

Next let's create dataset "Customers" based on SQL query. For this we use button located on bar of the "Data Browser" window. Using "Datasourse" dialog let's enter data source parameters.



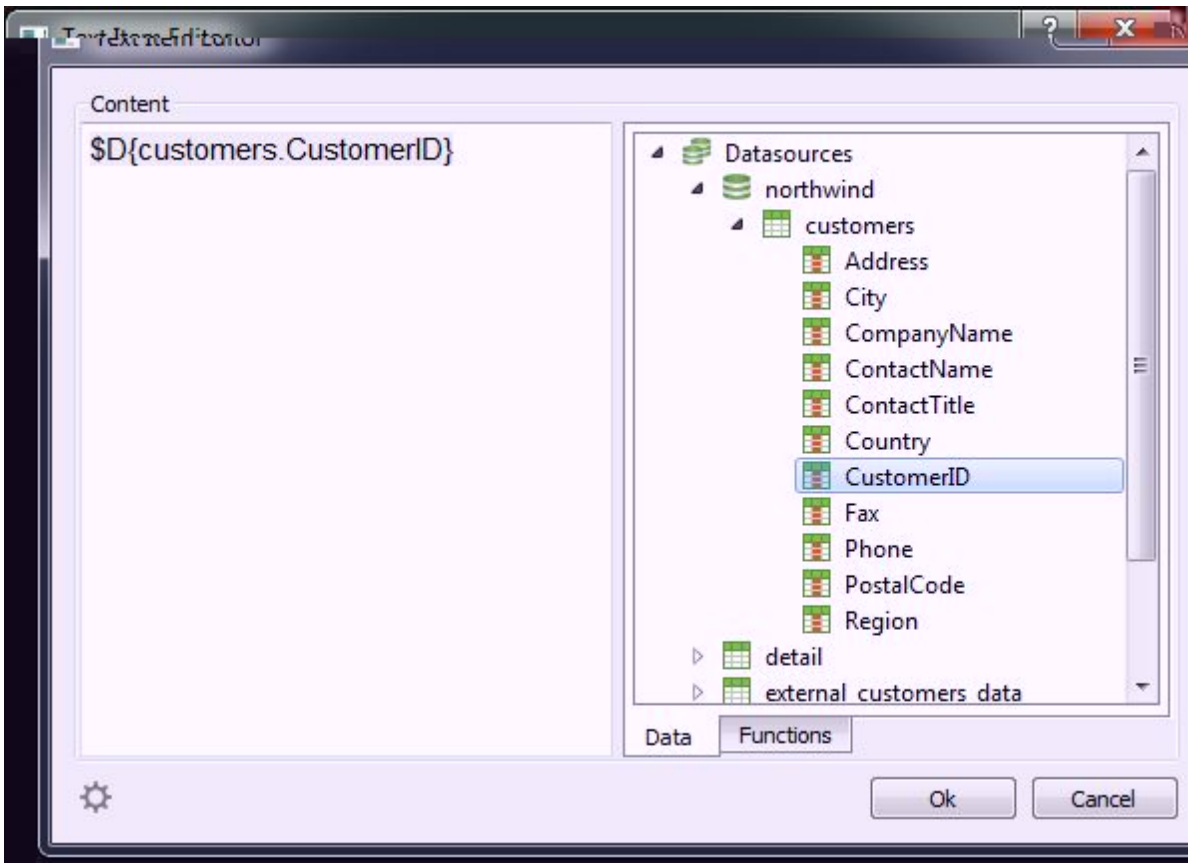
Connection - connection name (use formerly created connection "northwind")
Datasource Name - dataset name (we set "customers")
SQL - SQL query text (Select * from customers)

Now we can proceed to a report form creation. Using button (Insert band) we add a "Report Header" band to the report and place a "Text" object with text "Client list". Next we add "Data" band. Using ""Object Inspector" window we set "datasource" property to "customers".



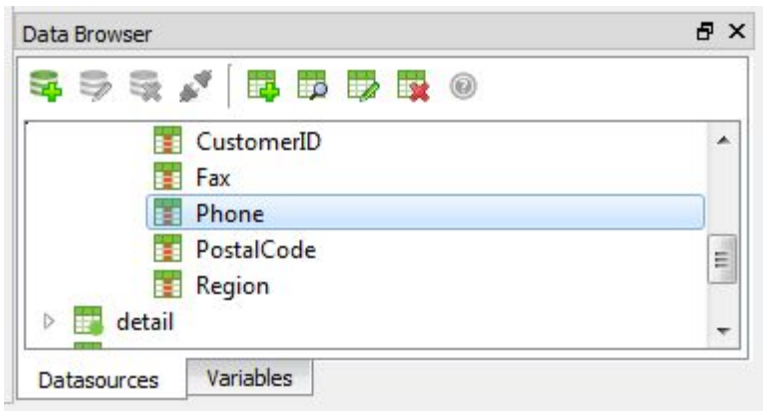
Now let's place four objects on the band, which will display client number, his name, phone and fax. Let's make it in two different ways to demonstrate powerful LimeReport designer capabilities. First text object we place on the band and print in it a text "$D{customers.CustomerID}". This is the most uncomfortable way as we have to print field reference manually and we can make a mistake. To

simplify such a reference insertion into a text a data source tree located in a right part of a text object editor can be used.

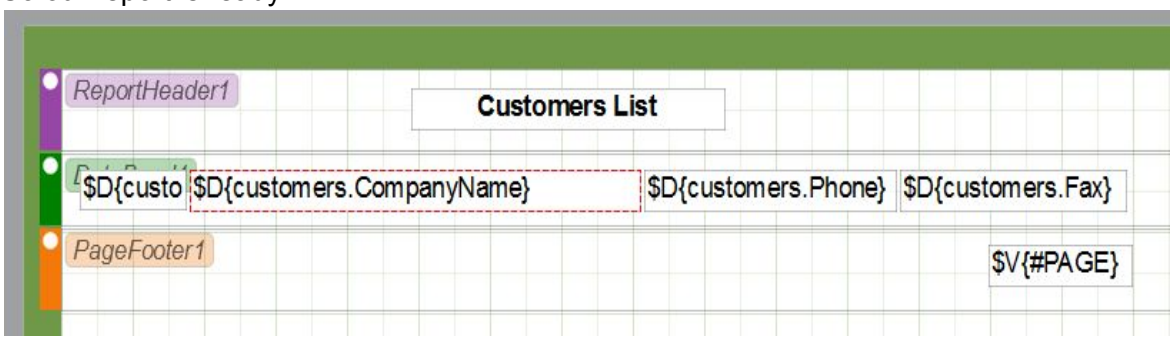


Let’s open tree nodes and find a proper field. Double click on it. A needed expression $D{customers.CustomerID} will be inserted in a current cursor position at the left side of the editor.

A second way - drag&drop of a necessary field from a "Data Browser" window into a report. This is the simplest and obvious approach. Capture a field "Phone" by a mouse and drag&drop it on a band.



So our report is ready



Press preview button and look at the result.

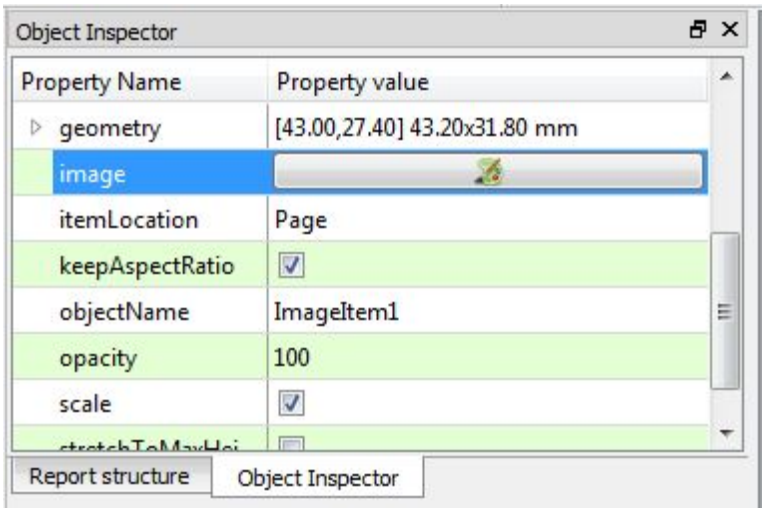
Customers List

ALFKI	Alfreds Futterkiste	030-0074321	030-0076545
ANATR	Ana Trujillo Emparedados y helados	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Taqueraa	(5) 555-3932	
AROUT	Around the Horn	(71) 555-7788	(71) 555-6750
BERGS	Berglunds snabbkap	0921-12 34 65	0921-12 34 67
BLAUS	Blauer See Delikatessen	0621-08460	0621-08924
BLONP	Blondel pare et fils	88.60.15.31	88.60.15.32
BOLID	Balido Comidas preparadas	(91) 555 22 82	(91) 555 91 99
BONAP	Bon app	91.24.45.40	91.24.45.41
BOTTM	Bottom-Dollar Markets	(604) 555-4729	(604) 555-3745

2.6 "Image" Object

Next object to consider is an "Image" object. It is also quite often used in reports. Using this object one can insert a company logo image, a photo of an employee or any other graphical information.

Let's consider capabilities of this object. Create an empty report and place an "Image" object on a report page. In "Object Inspector" select an "image" property and push button to select and load an image.

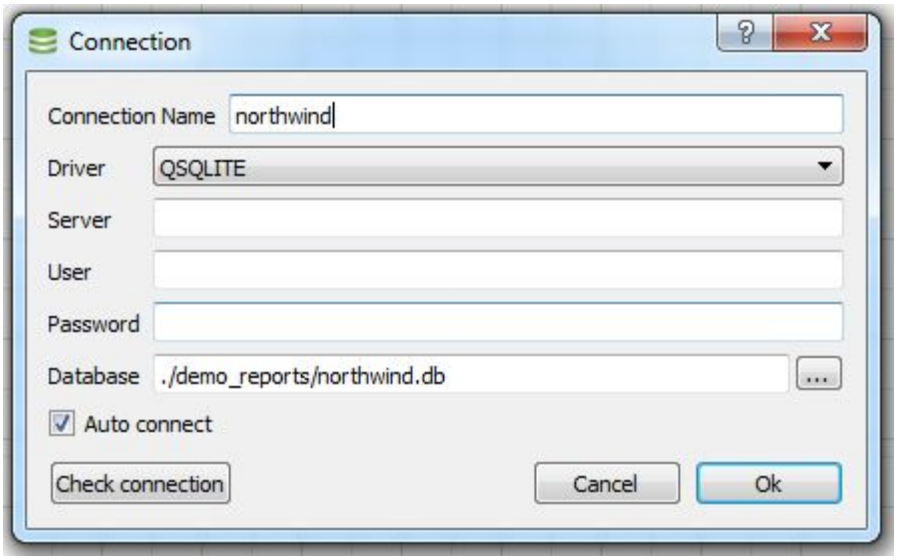


In "Object Inspector" we can also observe the following options: autoSize, scale (enabled by default), center (enabled by default), keepAspectRatio (enabled by default). Enabling "autoSize" option we see that the object took the size of the inner image. Sometimes such a capability is useful, if we need to display images of different sizes. By default this option is disabled what is appropriate for most of the cases. Option "scale" enabled by default and causes an image to stretch inside of an object. Change image sizes by mouse and you will see that image size corresponds to object size. If the option is disabled, then an image will be displayed with original sizes. This behavior differs from "autoSize" option in that object sizes are not adjusted to an image size, that is an object can be made bigger or smaller than an image. "center" option allows to center an image inside of an object. "keepAspectRatio" option is enabled by default and performs a very important task: it prevents from image scale distortion when object sizes change. This option only works in a pair with "Scale" option. When an object size changes a drawn circle remains a circle and does not turn into an ellipse. At that a stretched picture fills not a whole inner object space but only a part necessary for proper picture displaying with a correct image scale. When the option is disabled, a picture fills all the object space and if object sizes do not correspond original image scale the picture distorts.

2.7 A report with images

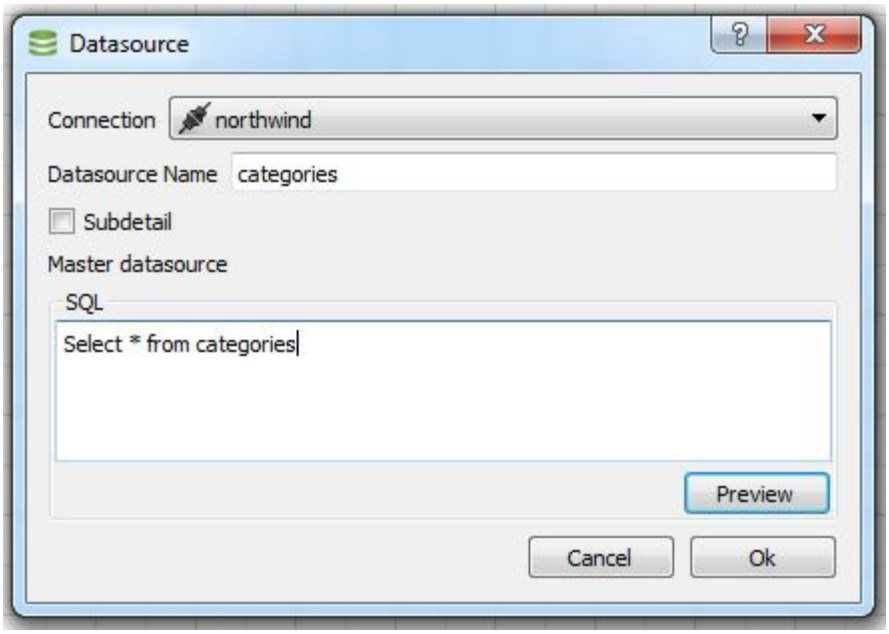
"Image" object as well as other LimeReport objects can display data from a database. Linking an object to a necessary database field is performed using "datasource" and "field" properties in the object inspector. In contrast to a "Text" object this is the only way to hook up data to an object. Let's demonstrate all abovementioned with a report example that will contain goods category images along with their titles. For this we need again demonstration database "nortwind" coming with a LimeReport suite. Let's open a designer and push a button "New Report" to create an empty template.

Let's link a dataset to the report in the window. To this end we use button (Add database connection), located on a bar of a "Data Browser" window. In a shown "Connection" dialog below let's set connection parameters.



Connection Name - connection name
Driver - database driver (SQLite)
DataBase - database path
AutoConnect - automatically connect after creating connection description and report loading.

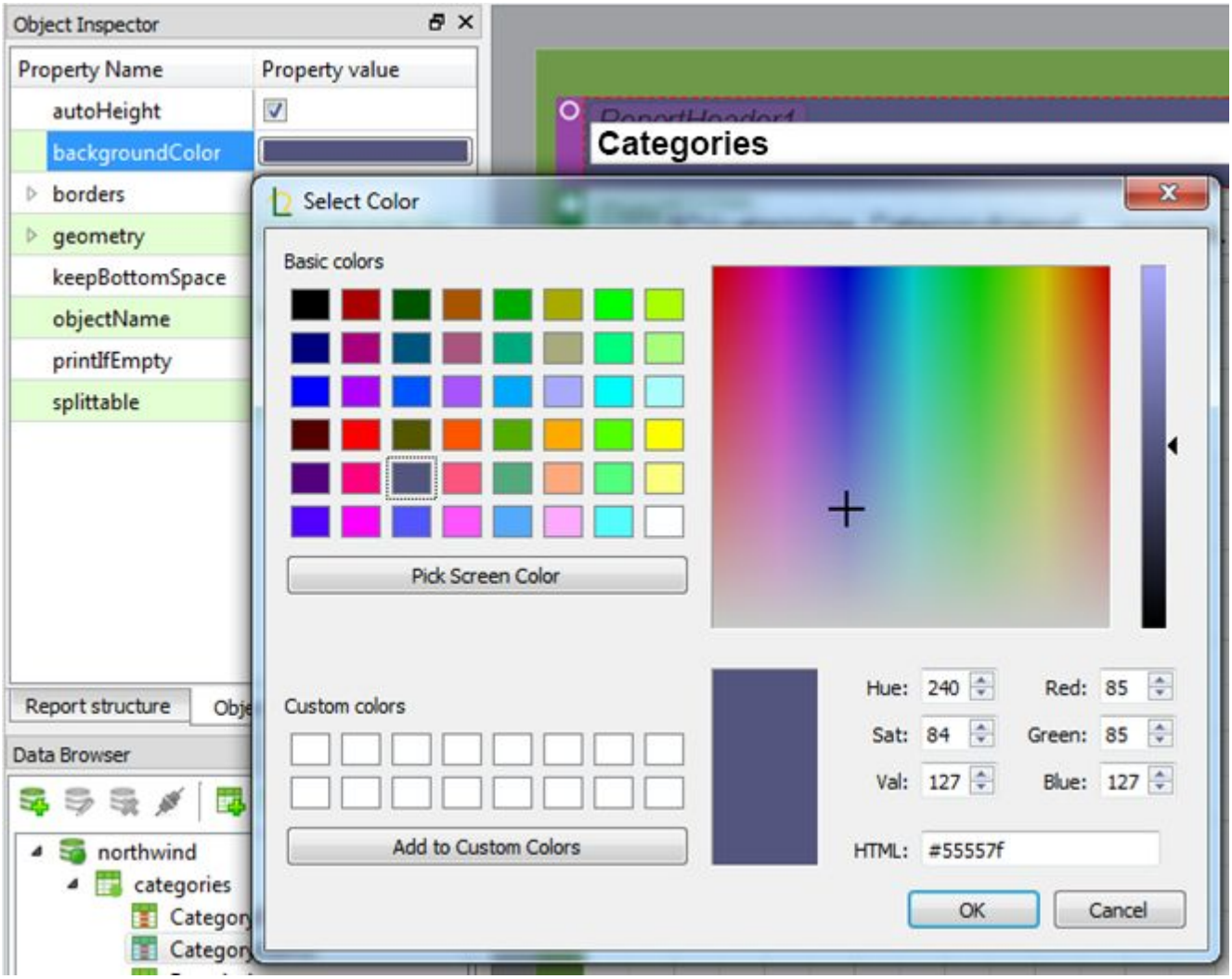
Next let's create dataset "Categories" based on SQL query. For this we use button located on a bar of a "Data Browser" window. Using "Datasourse" dialog let's adjust data source parameters.



Connection - connection name (select above created "northwind")
Datasource Name - data source name (print "categories")
SQL - SQL query text (print "Select * from categories")

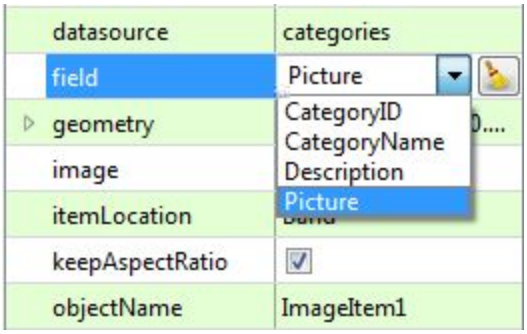
Now we can proceed to report form creating. We use (Insert band) button to add "Report Header" band to the report and place "Text" object with the text "Categories". Next we select a

band background color using backgroundColor property of the band object.

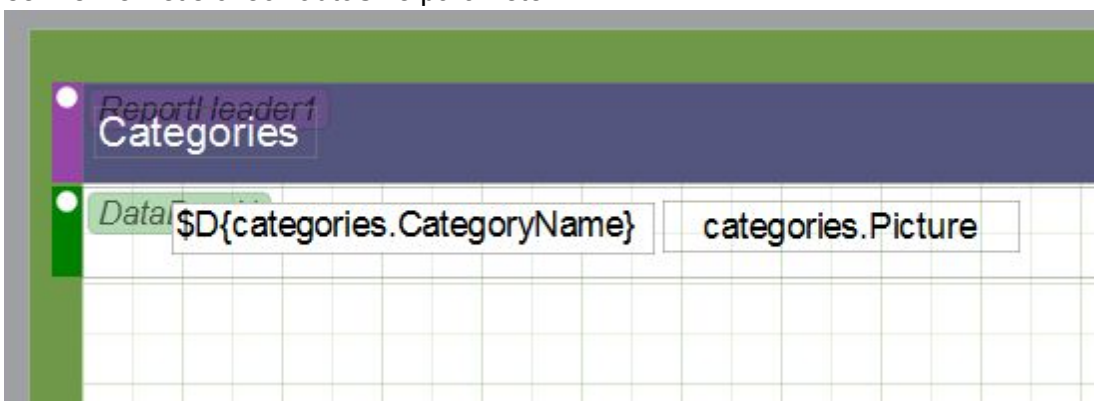


As we can observe the “Text” object is very notable against band background. To correct this we will change fontColor and backgroundMode properties of this object. Let’s set “backgroundMode” to “TransparentMode” and fontColor to while. Thus we get a white title on a transparent background.

Next we add “Data” band. Using “Object Inspector” window we set “datasource” property to “categories”. Let’s place "Text" object on the band and link it to a field $D{categories.CategoryName} with one of the approaches described above. Nearby we place "Image" object and link it to a "Picture" field.



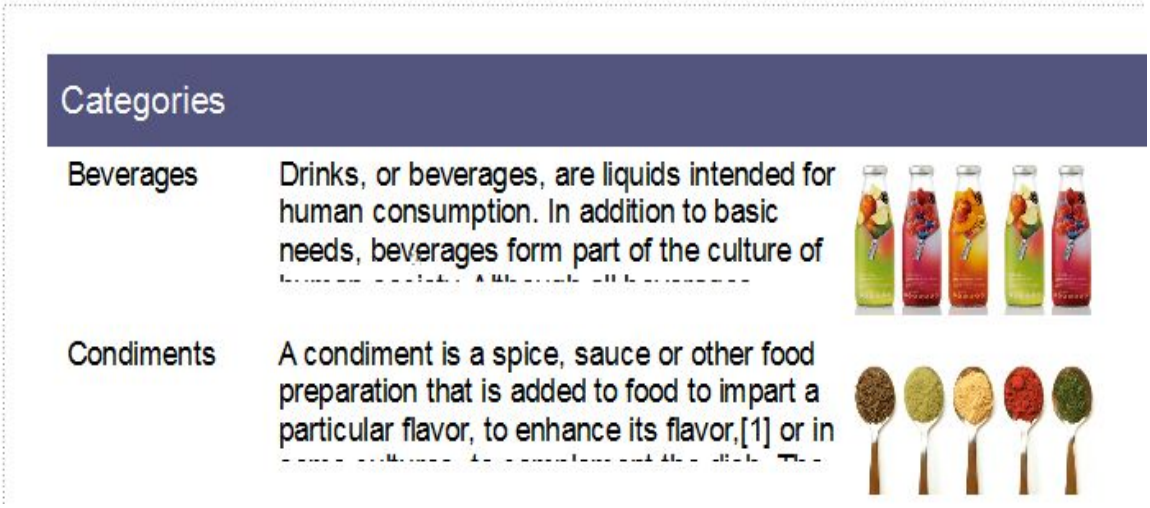
For this end in “Object Inspector” let’s adjust properties: datasource = “categories”, field = “Picture” Let’s remind that both these properties of "list" type, so needed values can be selected using a mouse. As well let’s check autoSize parameter.



That is it. Report is ready.

2.8 Displaying multiline text

Let’s come back to the previous example with categories. There is a "Description" field in a table “categories” which contains a description of each category. Let’s refine our report by adding this field to it. At a first sight everything is simple: we add a “Text” object on a data band, link it to the field "Description" and set object size. We run the report and see that what we have got is not exactly what we expected:



However LimeReport has merely done what we has asked it to do. "Description" field contains a multiline text, a size of which can vary. But our "Text" object displaying information from this field has a fixed size. Thus some lines did not fit into an object and were cut. What to do in this situation? We can of course tune object size with a margin or decrease a font size. However it will bring to an ineffective space usage on a page: some categories have a long description while others a short one. LimeReport has capabilities allowing to solve this issue. We speak about a band capability to select its height in such a way so that it can hold all its objects. For this end we just need to check an "autoHeight" property. However that is not the end of a story - an object with a long text should be capable to stretch itself. "Text" object can do that. An object can automatically select its height or width to completely hold all its text. “autoWidth” and “autoHeight” properties serve this purpose. "autoWidth" property selects an object width in such a way to hold all lines without hyphenation. This mode is useful when an object contains a single text line. "autoHeigh" property allows to select an object height so that all the text is within an object boundaries. Object width at that does not change. Check “autoHeigh” property for an object containing “Description” field. Check an "autoHeigh" band property as well.

Let’s run the report and make sure that now everything works as expected.

Categories

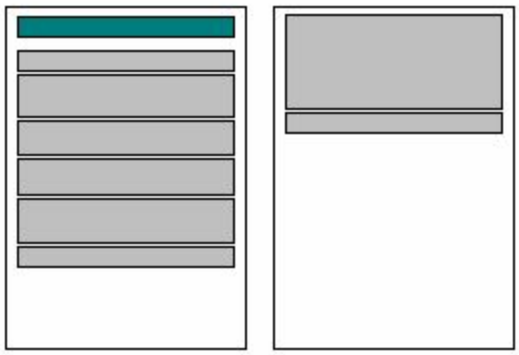
Beverages	Drinks, or beverages, are liquids intended for human consumption. In addition to basic needs, beverages form part of the culture of human society. Although all beverages, including juice, soft drinks, and carbonated drinks, have some form of water in them, water itself is often not classified as a beverage, and the word beverage has been recurrently defined as not referring to water . An alcoholic beverage is a drink containing ethanol, commonly known as alcohol, although in chemistry the definition of an alcohol includes many other compounds. Alcoholic beverages, such as wine, beer, and liquor, have been part of human culture and development for 8,000 years. Non-alcoholic beverages often signify drinks that would normally contain alcohol, such as beer and wine but are made with less than .5 percent alcohol by volume. The category includes drinks that have undergone an alcohol removal process such as non-alcoholic beers and de-alcoholized wines.
Condiments	A condiment is a spice, sauce or other food preparation that is added to food to impart a particular flavor, to enhance its flavor,[1] or in some cultures, to complement the dish. The term originally described pickled or preserved foods, but has shifted meaning over time.[2]

We see that while building a report LimeReport fills objects with data, stretches objects with enabled "autoHeigh" property and then selects a band height to hold all the objects. If a band "autoHeigh" option is disabled then height tuning is not performed and band is displayed with a height that was set in a designer. If we try to disable this option we will find that objects with long text still stretch but bands do not which also leads to a text cut.

2.9 A data split

Let’s notice one feature of the report with categories: on some pages at the bottom there is much free room. Why does it happen? When a report is built LimeReport kernel fills a free page space with bands. After displaying each band a current position shifts lower and lower. When LimeReport discovers that there is not enough space for a next band (its height is greater than the height of the remaining page space), then a new page is created and the output of bands is continued on it. And so on until a dataset has records.

Our report just contains an object with a lot of text so band height is quite large. And if a large band does not fit into a page then it goes to the next page but at the bottom of the page there is a lot of empty space. It can be seen on the following picture:

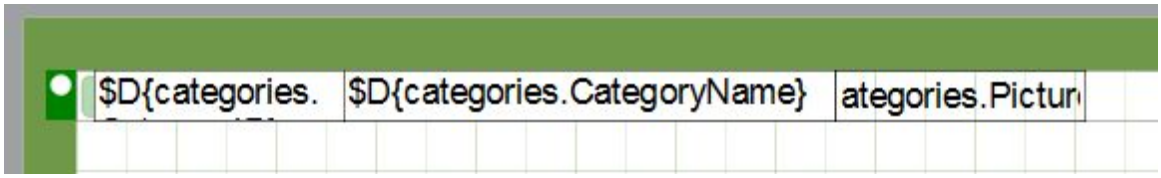


To use a paper effectively let’s use LimeReport capability to split a band into parts. All we need is to enable "splittable" option of band "Data". We can see that a free room at the bottom of the report pages considerably diminished:

It is worth noting that the split algorithm does not provide 100% quality report. So use it with caution.

2.18 Displaying data in the form of a table

It is often needed to display a report as a table with borders. One of examples of such a report is a price list. To build such a report with LimeReport one should just enable borders of “Data” band objects. Let’s consider several variants of borders using test report example. We create a report of the following view:



Let’s place band objects edge-to-edge and reduce a band height to a minimum size. First and the simplest type of such a table is with all borders. For that end we should enable all border lines:

1	Beverages	
2	Condiments	
3	Confections	
4	Dairy Products	
5	Grains/Cereals	
6	Meat/Poultry	
7	Produce	
8	Seafood	

Next border type – only horizontal or vertical lines – can be made similarly, horizontal or vertical object borders are enabled.

1	Beverages	
2	Condiments	
3	Confections	
4	Dairy Products	
5	Grains/Cereals	
6	Meat/Poultry	
7	Produce	
8	Seafood	

All abovementioned examples contain bands with a constant size. But how to display a table if a band is stretchable? Let’s show this by example. We enable autoSize property of an “Image” object.

In this case a band height will be selected depending on an image size. We will obtain a report of the following view:

1	Beverages	
2	Condiments	
3	Confections	
4	Dairy Products	

A little not what we need – we would like for adjacent object borders to stretch as well. LimeReport solves this issue easily. For building such reports it is sufficient to enable “strechToMaxHeight” property of all the objects that should be stretchable. At that LimeReport kernel calculates at first maximum band height, then stretches objects with the enabled property to the bottom edge of the band. As a border moves with an object the report view changes as a result:

1	Beverages	
2	Condiments	
3	Confections	
4	Dairy Products	